

Aspects of Computing and Software in a HEP experiment

Simona Rolli

Introduction

Over the past ~12 years I have been involved in several software projects while being a member of the CDF collaboration at FNAL



- Over the lifetime of CDF II, the needs of the experiment changed:
 - ♦ The collaboration decided to rewrite all of our software in C++ ;
 - ♦ We faced the task of organizing very large amount of data, that needed to be readily accessible to the whole collaboration
 - Some of the GRID concepts did not even exist in 1997;
 - ♦ We commissioned the detector with new hardware and software
 - more than a simple upgrade, more like a new detector
 - ♦ We finally reached a steady state
 - Most of the tasks are now being “automatized” to compensate for shortage of manpower

Outline of the talk

- **A new beginning: from Fortran to C++**
 - ◆ Data handling solutions based on OO databases
 - ◆ THE CDF Hybrid solution
- **Data Access and Analysis**
 - ◆ Standard ntuples and unified analysis tools
- **Simulation and Reconstruction**
 - ◆ Trigger Simulation at CDF
 - Offline tool
 - Online tool
- **The Steady state**
 - ◆ Automatic Tools for performance and ID calculations

A new beginning....



At CHEP 1994 a new Word began spreading in the HEP community....



One of the results of the conference was the decision in the few subsequent years on the part of many experiments to write their software in an object oriented fashion

CDF and D0 found themselves in a frenzy being in the middle of a major upgrade from Run I to Run II.

CDF decided to proceed in the following way.....

CDFII Software Setup

- **Online systems** would be using Java for boards control GUI and Run Control (C code is mostly used to talk to the boards, it easily translated in C++ for emulation/simulation);
- ♦ **Offline software**(simulation and reconstruction) would be totally written in C++;
- ♦ The **data handling** (very large volume of data) would be based on the use of the ROOT data format coupled to a relational databases to make up for catalogue functionality (file-based catalogue).
- ♦ The **analysis framework** choice would be left to the user (HepTuple interface)
 - ♦ Eventually a few standard ntuples became the tool of choice for most of the collaboration physics analysis groups

Data handling challenges

Estimated run II needs for CDF/D0

Category	Parameter	DØ	CDF
DAQ Rates	peak rate	53 Hz	75Hz
	equivalent DC rate		
	averaged over the year	20 Hz	28Hz
	average event size	250 kB	250 kB
	average data rate	5 MB/s	7 MB/s
	level 2 output	800 Hz	300 Hz
	maximum data logging bandwidth	scalable	80 MB/s
Data Storage	number of events	600 M/year	900 M/year
	RAW data	150 TB/year	250 TB/year
	reconstructed data STA(DØ)/DST(CDF)	75 TB/year	135 TB/year
	physics analysis data DST(DØ)/PAD(CDF)	50 TB/year	79 TB/year
	μDST	3 TB/year	-
	total data volume	280 TB/year	464 TB/year
CPU:	reconstruction/event	1000-2500 MIPS-s/event	1200 MIPS-s/event
	reconstruction	34,000 - 83,000 MIPS	56,000 MIPS
	analysis	60,000 - 80,000 MIPS	90,000 MIPS
Software	event database	required	required
	process control	required	required
	storage management	required	required
	secondary databases	required	required

Table 1: Summary of estimated Run II needs for CDF and DØ. Storage needs are on a yearly basis.

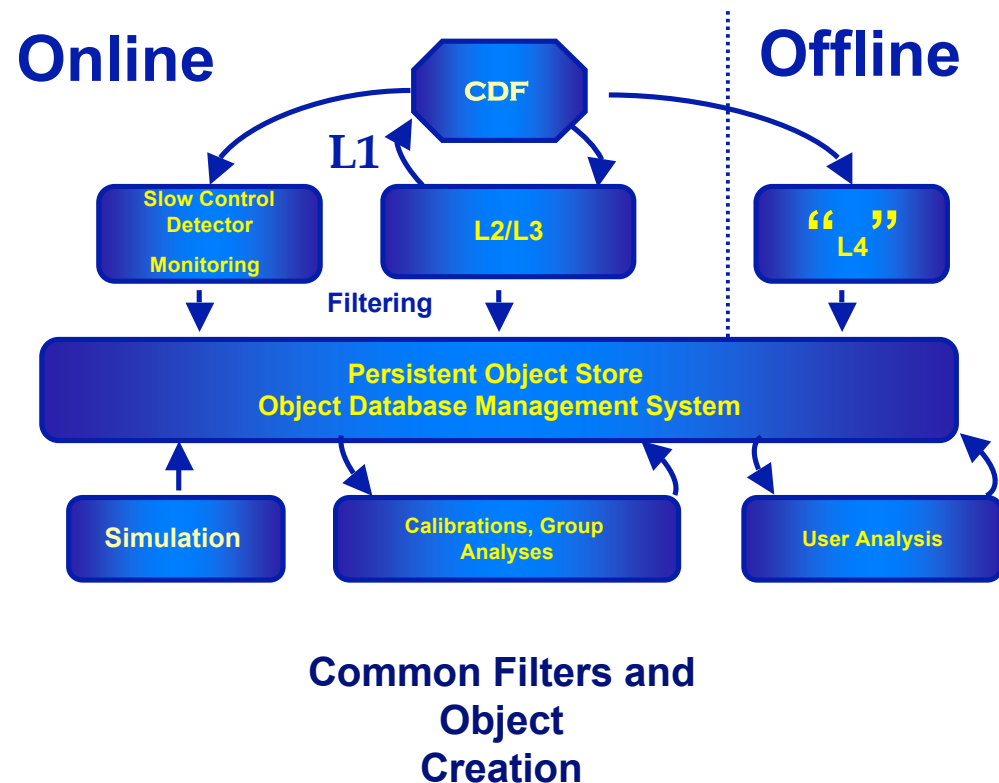
- **Data access questions:**
 - ♦ **what event we look at:**
 - Data
 - Calibration and support
 - simulation
 - ♦ **how to select events**
 - ♦ **what processing steps needed for event selection/event analysis**
 - ♦ **how many time and how often data are accessed?**
 - ♦ **Which piece of the events are needed?**

S. Rolli et al. Objectivity WorldView, S. Clara, May 1997

OODBMS Data handling

One of the proposals for CDF was based on the solution being worked out for the LHC experiments

- Split the event information in different parts, residing on different media, depending on the access patterns, maintaining associations
 - ♦ AOD on disk
 - ♦ ESD on tape
 - ♦ RAW on tape
- Use direct access to access needed information
- Use an OODBMS as object manager (off-the-shelf system)
- Possible serious drawbacks when using serial media (tape)



K. Karr, S. Rolli, K. Sliwa CDF data management in Run-II and Objectivity ODBMS
CDF/ANAL/CDF/PUBLIC/4201, 1997

CDF Data access and data handling

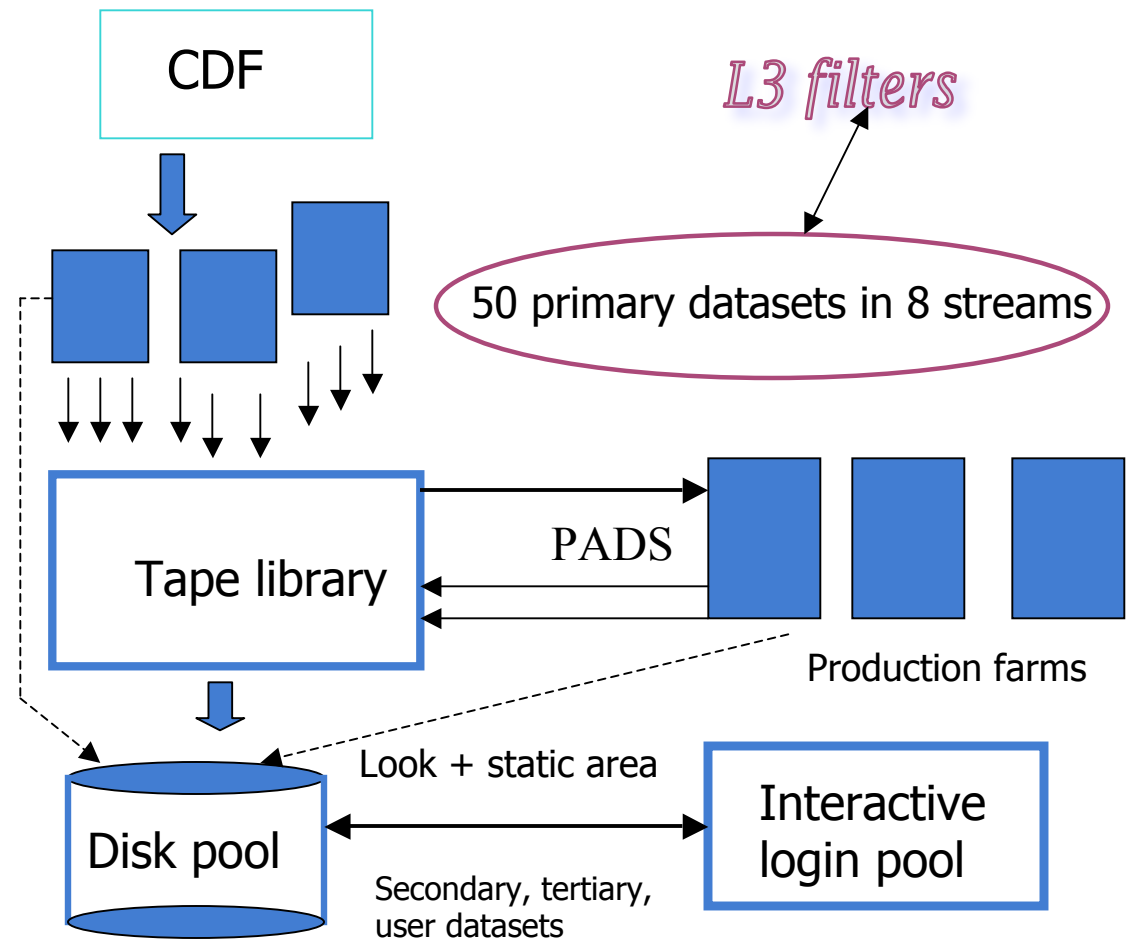
The solution CDF adopted was an hybrid one:

It uses **ROOT**, an HEP specific data analysis framework, as its data format (sequential), but builds an intermediate layer (Event Data Model) which allows OO functionality.

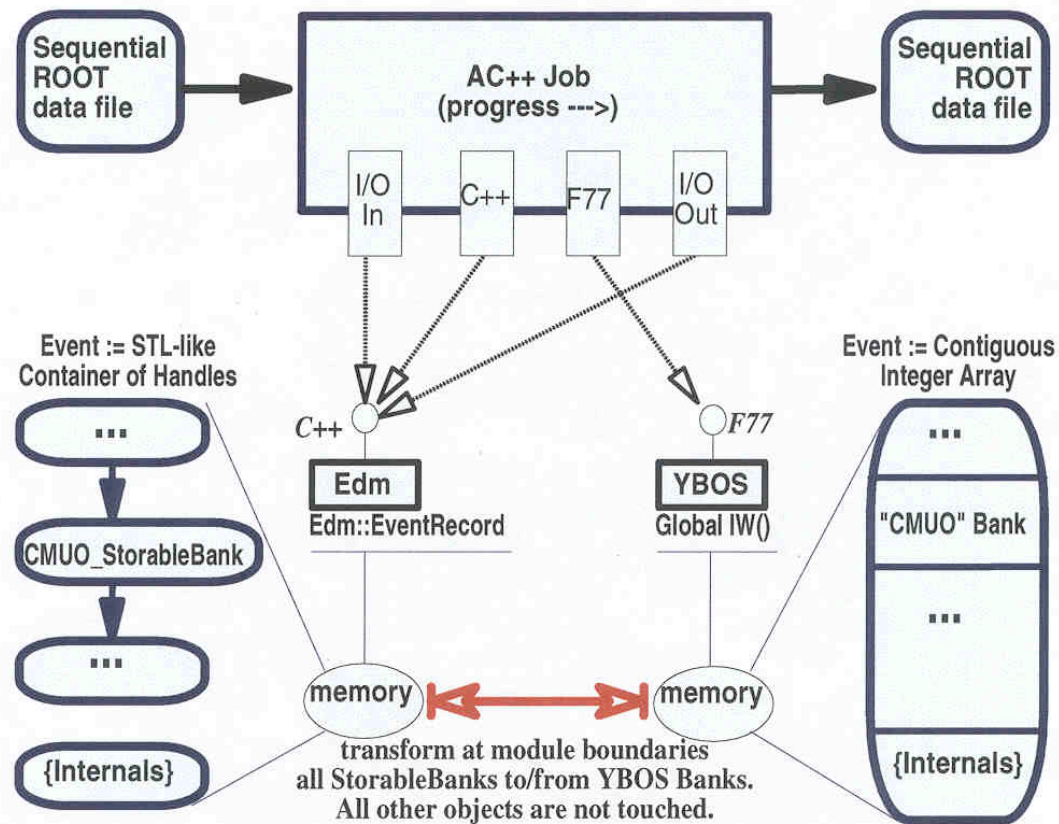
The data handling system makes use of a **relational database** to make up for catalogue functionality (file-based catalogue).

Data are kept on a tape robot and spooled on disk upon request.
No direct tape access

Data are also kept in an extremely highly compressed format (Physics Analysis Data, PAD's).



CDF Event Data Model

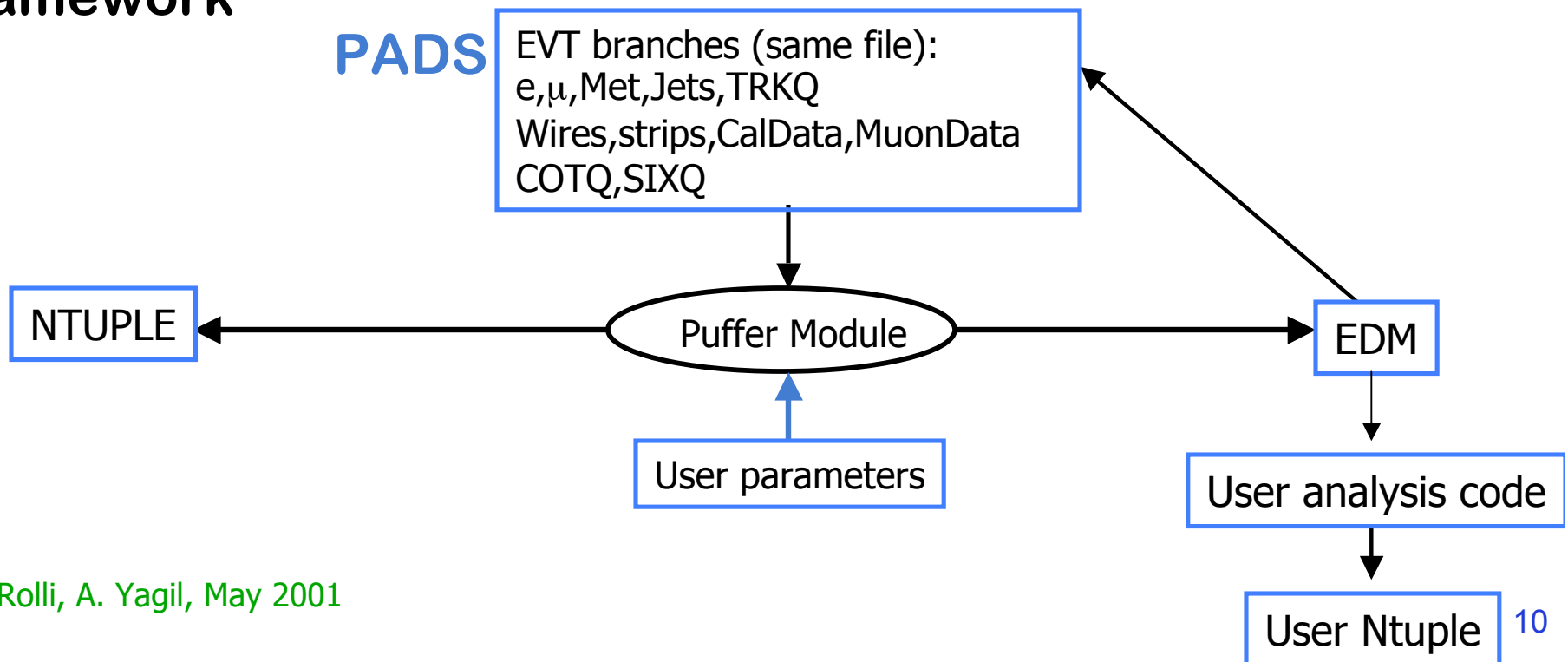


R. Kennedy, S. Rolli et al. CHEP 2000, Computing in high energy and nuclear physics* pg 442-44.

Data Analysis Model at CDF

PADS data files are not directly readable in the ROOT analysis framework

Data needs to be “puffed” using the EDM intermediate layer (dictionary description) to be used in an analysis framework



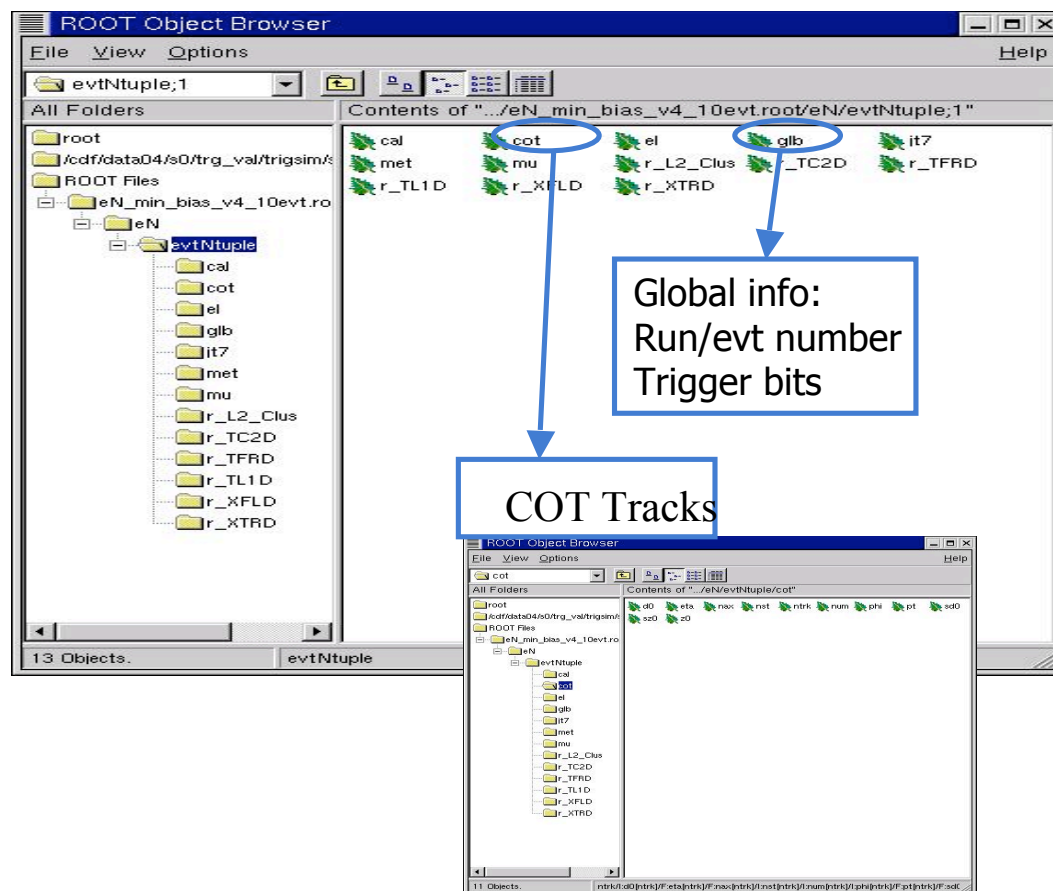
evtNtuple

The PADS Event information is translated into ROOT branches:

- High Level Objects
- Trigger Information
- Raw Data Information
- Simulated information

The User can add his/her own branch for specific analysis needs (derived quantities)

The ntuple is portable outside of the CDF software environment (the User laptop..)



<http://ncdf70.fnal.gov:8001/talks/eN/eN.html>

One of the analysis tools used in CDF

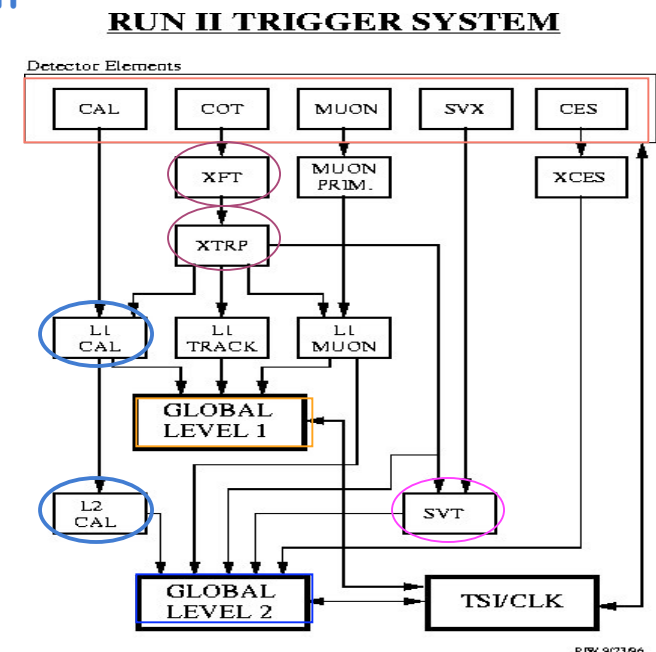
Trigger Simulation at CDF

- **TRGSim++** is a set of (C++) packages which emulate the L1 and L2 hardware trigger levels decision steps
 - ♦ offline tool to calculate rates and efficiencies;
 - ♦ online monitoring tool for the trigger boards.
- TRGSim++ modules run off detector raw data and produce emulated trigger data identical to real hardware data.
- Not a time-critical application, more a “static” emulation

- Trigger decision steps: A_C++ modules, packages:

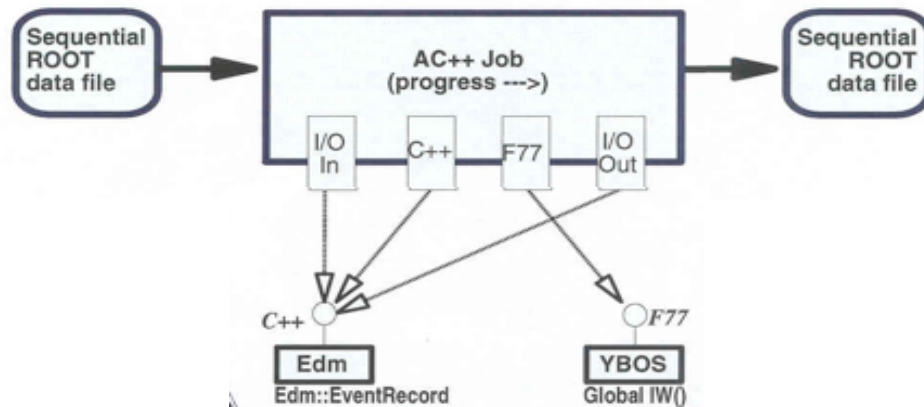
- ♦ CalTrigger
- ♦ MuonTrigger
- ♦ XFTSim
- ♦ SVTSim
- ♦ XTRPSim
- ♦ L2/L1GlobalTrigger
- ♦ TriggerMods
- ♦ TriggerObjects

Coordination of a group of about 30 detector experts



The Simulation Code

The simulation code has to adhere to the AC++ philosophy: Modules representing functions talk to each other via data structures (banks -part of the YBOS array, or storable Objects) contained in the EventRecord.



This is also the only part of the CDF simulation/reconstruction code which can reproduce bit by bit the digital hardware behavior. It is essentially code that translate in a higher language the behavior of the trigger boards.

The challenge is to write code which is modular and allows for as much recycling as possible of the online boards (C)code.

Another challenge was to get the “hardware” people to write C++ software, providing them with a general framework and infrastructure

- db access for instance.

Example: The Calorimeter Trigger

L1:

Trigger on electrons, photons, jets (object triggers),

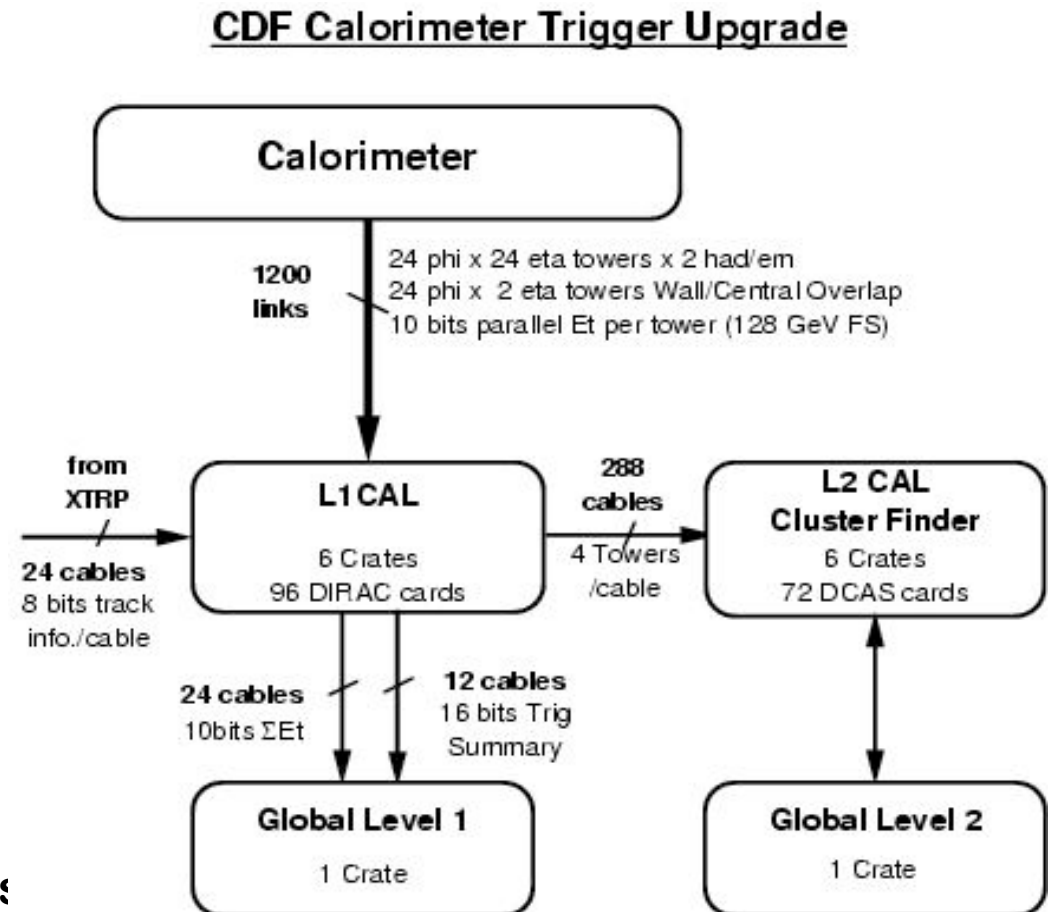
total event transverse energy and missing transverse energy (global triggers)

Object triggers : threshold applied to individual towers;

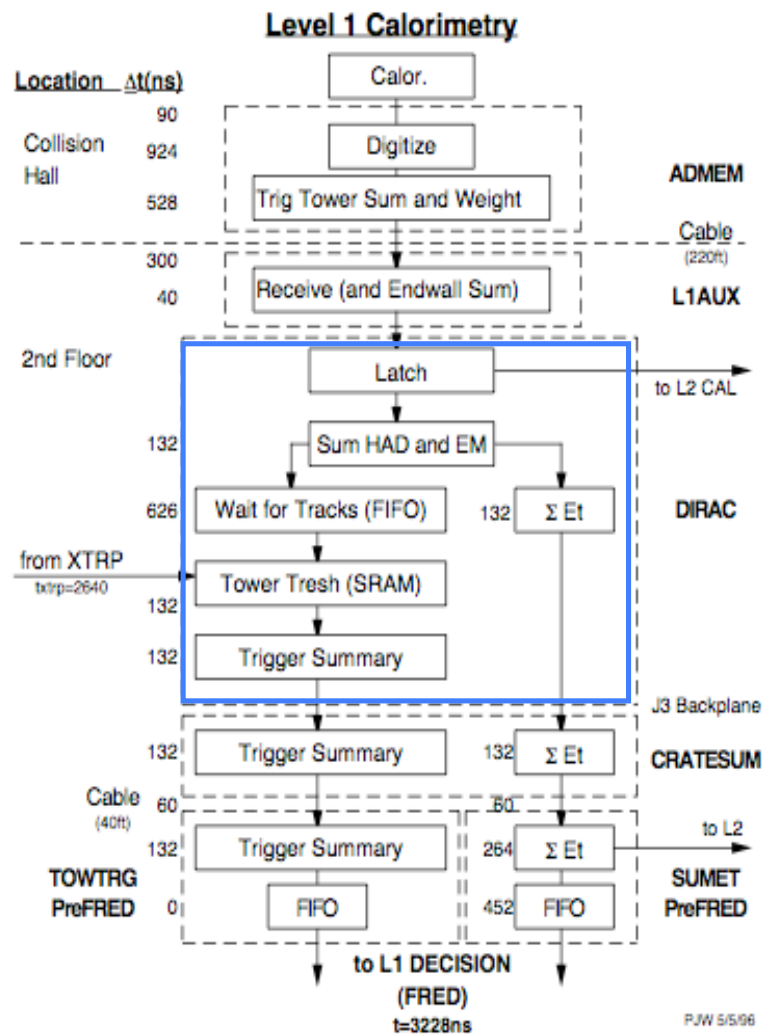
Global triggers : threshold applied after summing energies from all towers.

L2 :

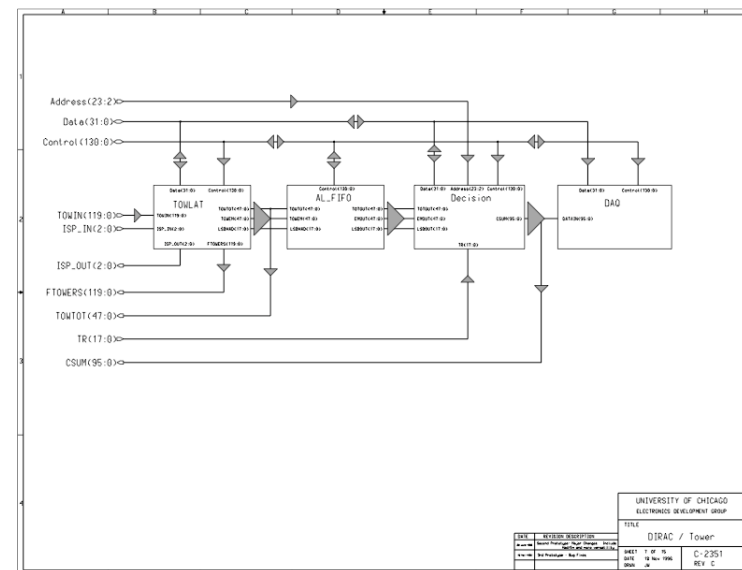
cluster finder and isolation sums



Example: L1 Calo Trigger



```
class DiracDat {
private:
    //inputs
    int etin[13]; /* the 13 input channels */
    int trin; /* 8 bits of tracking input */
    //outputs
    int output[12]; /* the front panel output to DCAS */
    int daq[4]; /* daq[i] i=word */
    int etsum; /* 8 et bits sent to cratesum */
    int trsum; /* 16 trigger bits sent to cratesum */
    int slot /* bunch crossing number, slot number */
    int etaux[12]; /* the et out of the aux card */
    int towem[6]; /* the towlat output, 8 bits of em et */
    int lsbhad[6]; /* the towlat outputs for lsbhad(3) */
    int towtot[6]; /* 8 bits of total energy from towlat */
    .....
}
```



DIRAC crates

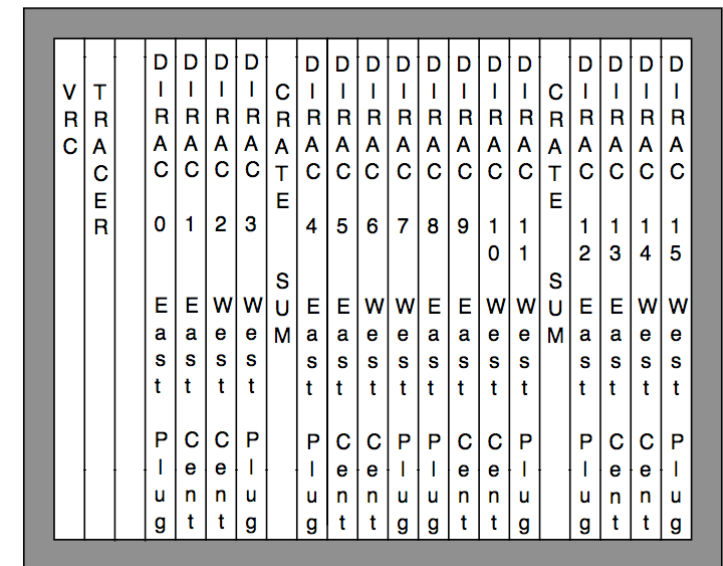
L1 Dirac boards receive their input from ADMEM (FE boards on the detector):
 - 10 bit word for each trigger tower ($15\phi \times 0.2\eta$)
 for a total of 24×24 towers

Each card processes 6 HAD and 6 EM towers from a quarter of a wedge in ϕ .
 Therefore each card receive 120 bits from the detector
 There are 16 DIRAC cards in each crate
 There are 6 DIRAC crates (for 576 Trigger Towers)

There is one Crate Sum board per eight
 DIRAC boards. CS receives an 8 bit Sumet
 and 16 bit trigger Summary Word from each
 DIRAC via a custom backplane

In the code we get the TDC data from the
 raw data banks, emulate the ADMEM PMT sums
 and weighting with routines and pass the
 energies to the Dirac board object, via
 the etin array of input channels

DIRAC VME Crate

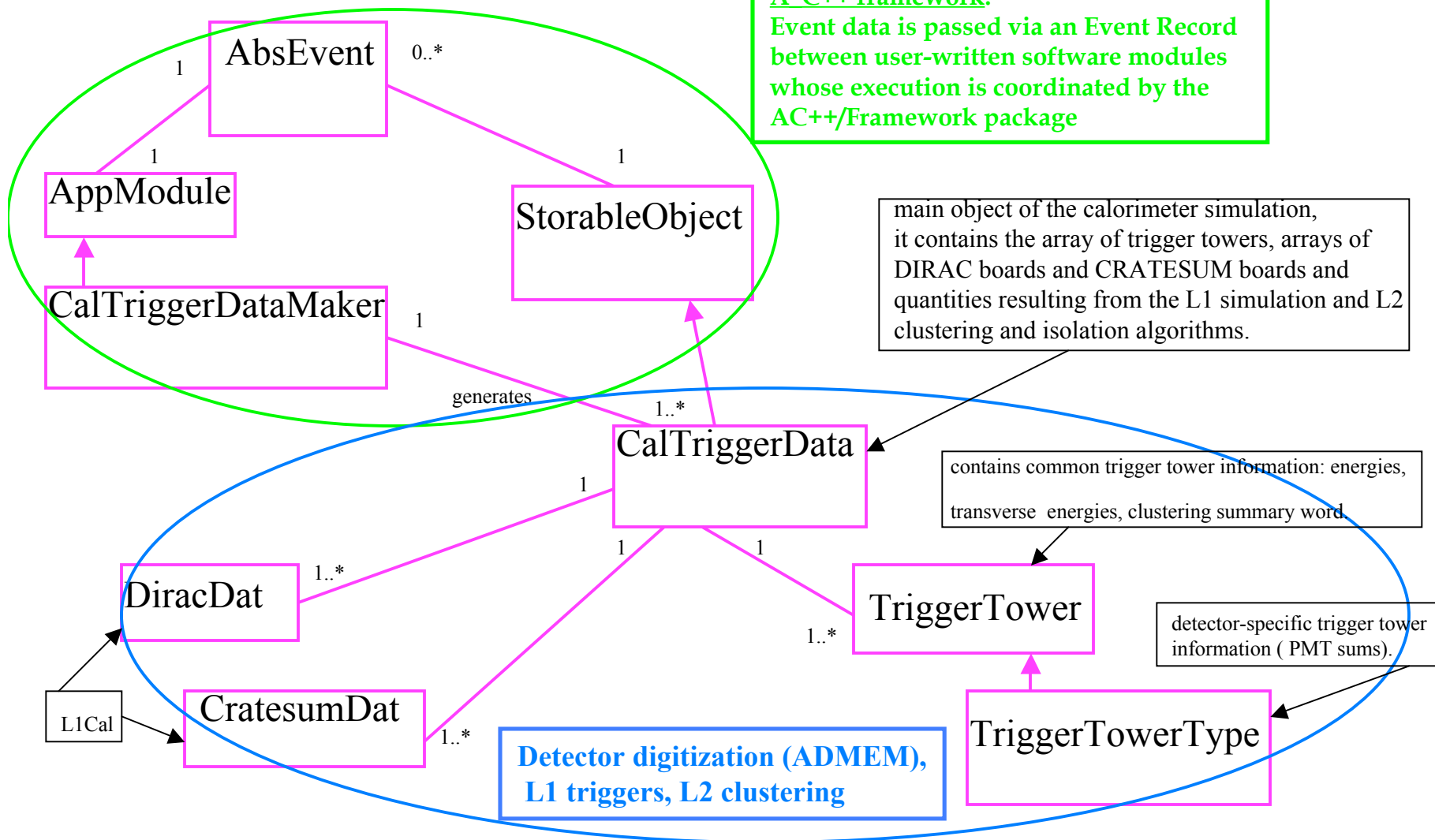


ϕ Wedge: 0 1 2 3 10

CalTrigger package

A C++ framework:

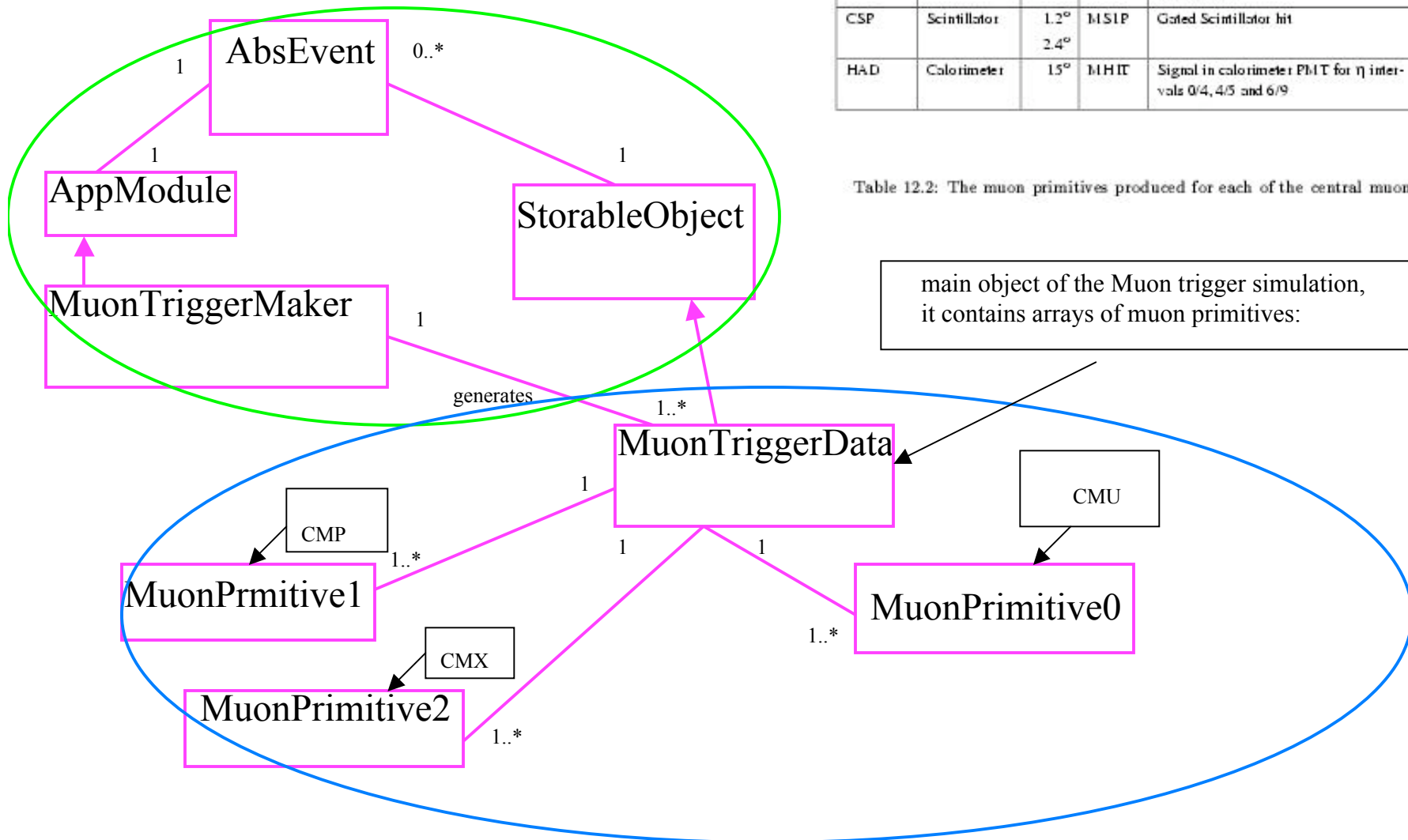
Event data is passed via an Event Record between user-written software modules whose execution is coordinated by the AC++/Framework package



MuonTrigger

Detector	Basic Unit	Unit	CARD	Algorithm Description	# Outputs
CMU	Wire pair	1.25°	MUIT	Hi, low P_t , plus a "lefover to 384ns" determined from differential timing	288 x 2 x 2 $\phi \pm \eta \pm P_t$
CMP	4 tube stack	0.6° 1.2°	MPIT	2 or 3 out of 4 hits for patterns from radial tracks	336 ϕ
CMX	Wire pair	1.25°	MXIT	Hi, low P_t , plus a "lefover to 384ns" determined from differential timing	288 x 2 x 2 $\phi \pm \eta \pm P_t$
CSX	Coincidence	15°/8	MS1X	Gated Mean Time from 1/2 overlapped scintillators	192 x 2 $\phi \pm \eta$
CSP	Scintillator	1.2° 2.4°	MS1P	Gated Scintillator hit	168 x 2 $\phi \pm \eta$
HAD	Calorimeter	15°	MHIT	Signal in calorimeter PHIT for η intervals 0/4, 4/5 and 6/9	24 x 6 $\phi \pm \eta$

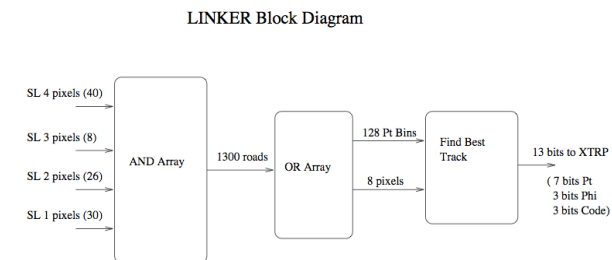
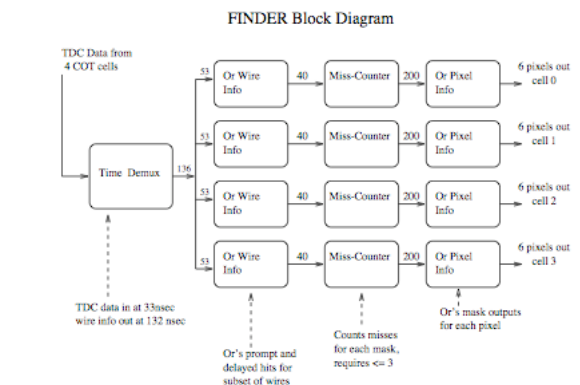
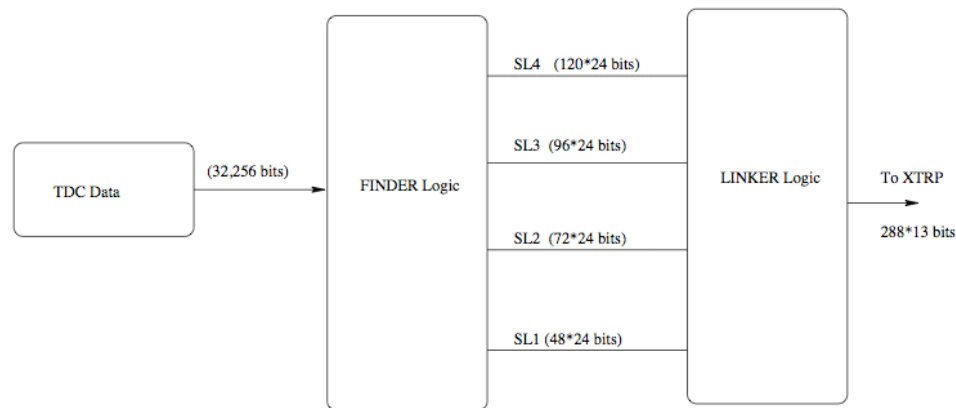
Table 12.2: The muon primitives produced for each of the central muon detectors.



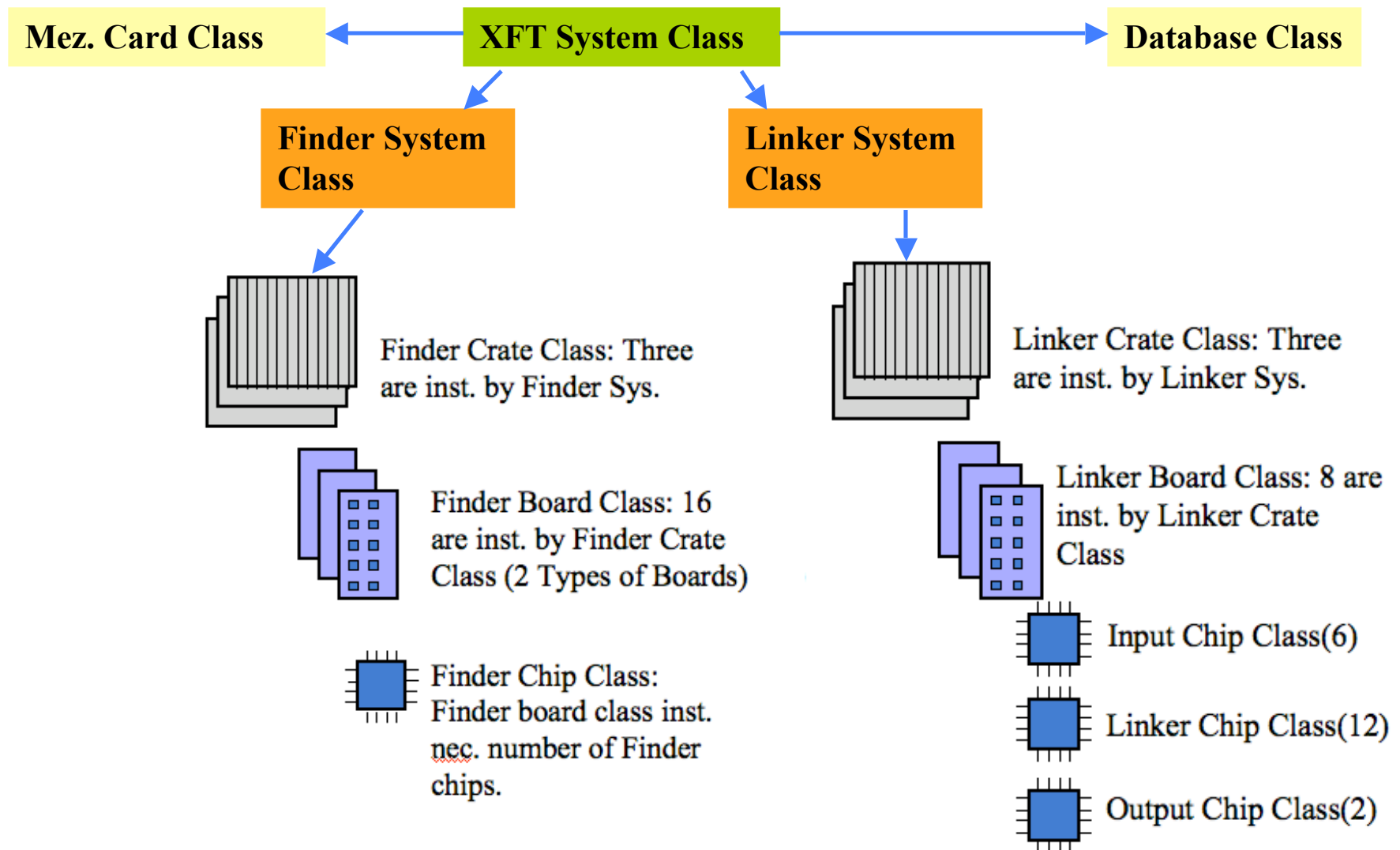
EXtremely Fast Tracker

Data from the Central Tracker (COT) are processed from each bunch crossing, and the result is available in time to be used in the L1 trigger decision.

- The processor works off hit data from the 4 axial layers of the COT. Data from each wire is classified as prompt or delayed (32K bits) depending on the maximum drift in the COT.
- Track identification is accomplished in two steps: the Finder and the Linker.
- Finder looks for high P_T track segments in the SL
- Linker searches for match among segments

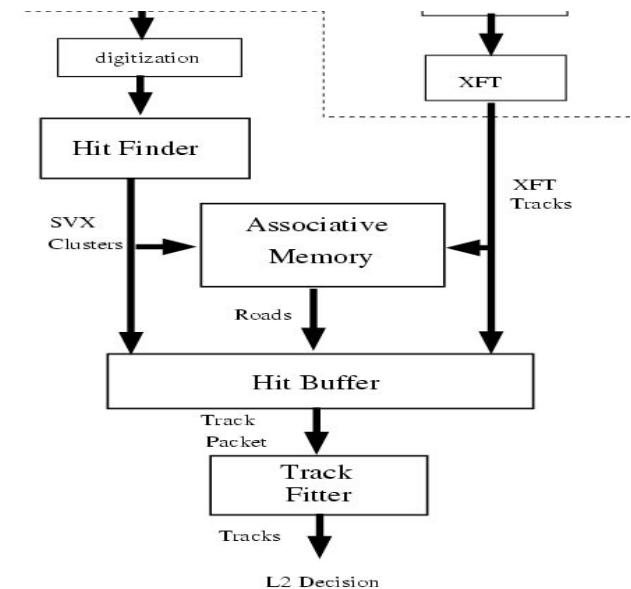


XFTSim



Secondary Vertex Trigger

CDF decided to use impact parameter information of the tracks in the trigger to detect secondary vertices as a tool to substantially increase the physics reach of CDF

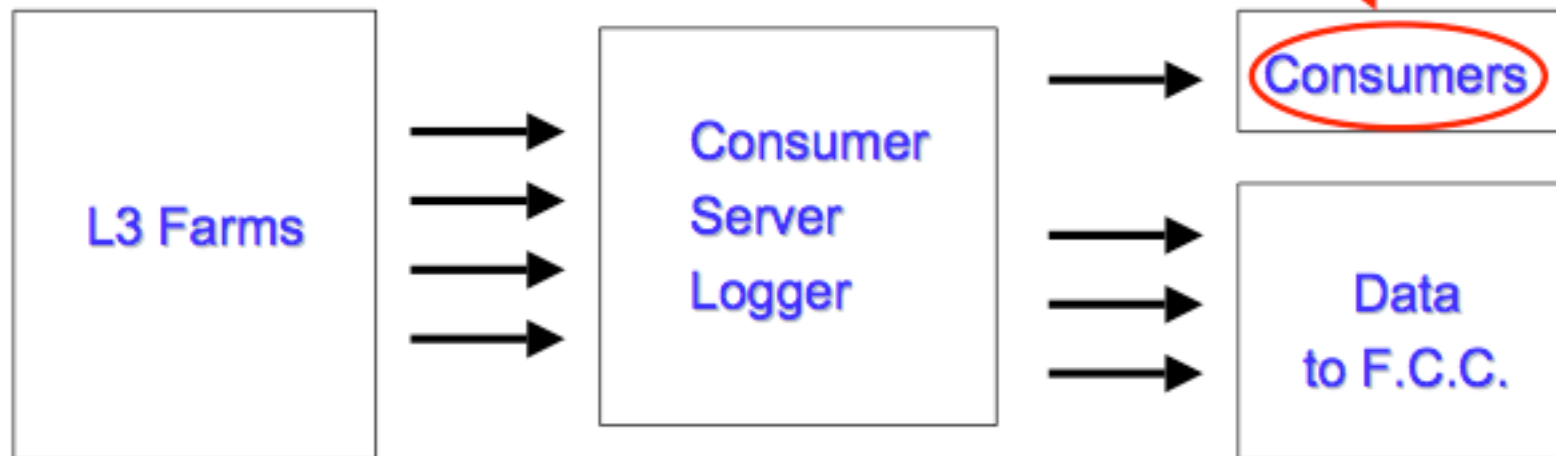


- Modular C code , with “board simulator” for each board
 - ♦ C code needed to *run the same simulators both from TRGSIM and in crate controller CPU’s for online diagnostic.*
- An A_C++ module provides the C simulators with Silicon raw data information and XFT reconstructed tracks. It writes out a simulated SVTD_StorableBank (identical to the hardware bank)

Monitoring the trigger online

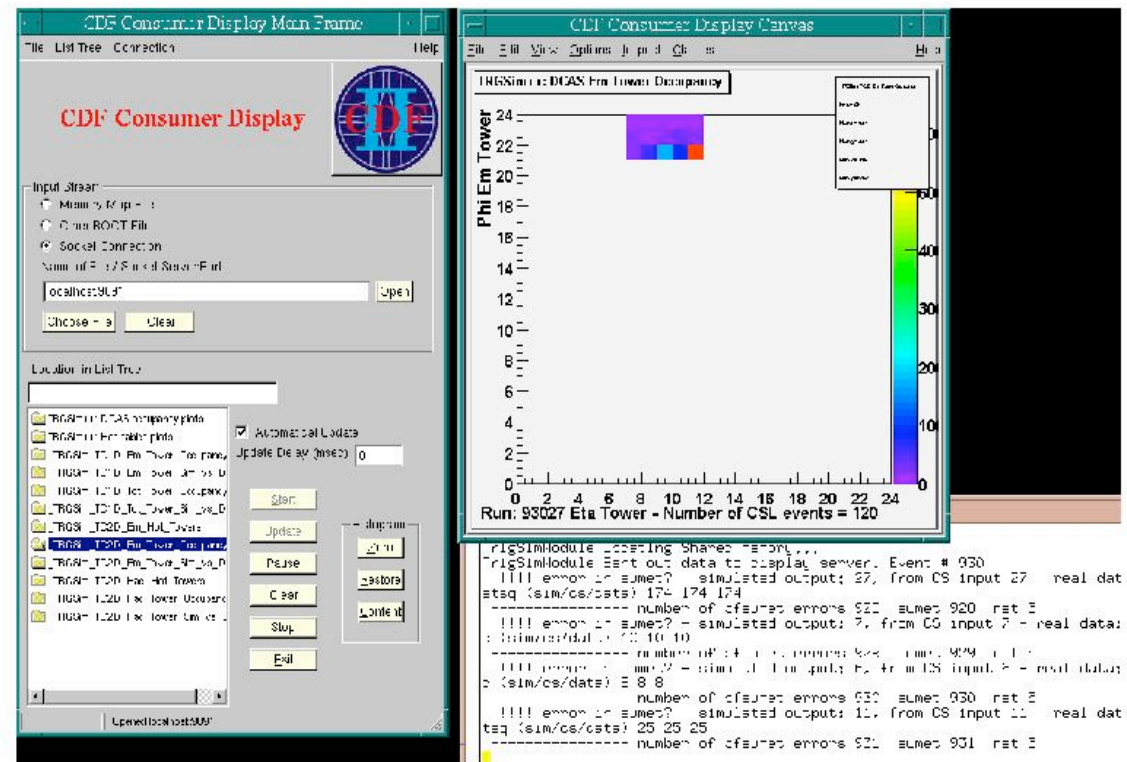


Generates plots, etc. to verify
quality of data being taken.



Monitoring the trigger online

- Each subsystem driver module contains **data monitors methods** which check word by word real and simulated banks, firing error messages on the error logger and on the standard output if discrepancy is found
- Summary files produced for every run, control on level of printing
- Easily coupled to the online ROOT-based Consumer Framework (HistoDisplayMain)
- Now also on the Web



The Steady State

- People are migrating to the LHC [and other experiments]
 - This is not new, started a long time ago
- CDF has taken many measures to mitigate the impact on the experiment
 - We have stabilized, streamlined and automated many tasks in operations and in physics analysis

Among these:

- Common Ntuples
- Calibration Tasks
- Efficiencies and Scale Factors

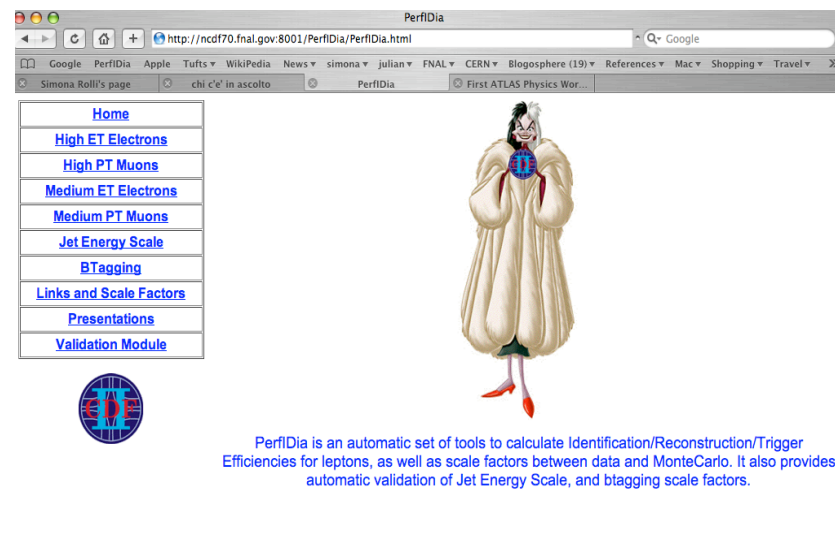
PerfIDia (Performance and ID instant answer)

The idea is to have much of analysis infrastructure at CDF running in an automatic way to guarantee smooth running in the final years of the experiment

Some aspects of all analyses are in common:

lepton ID efficiency,
reconstruction, trigger
Jet Energy corrections
B-tagging Scale factors
tau reconstruction

.....

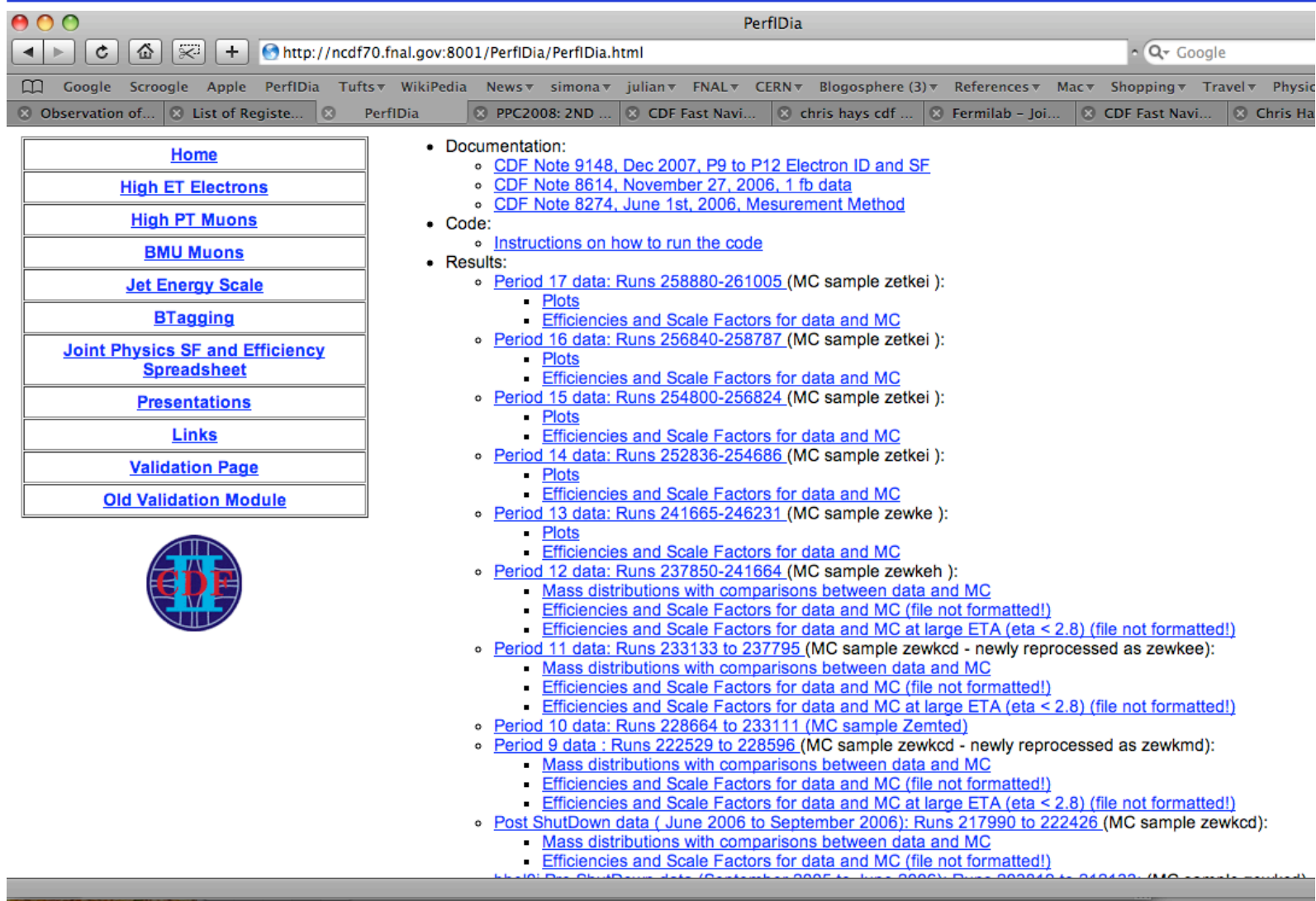


We are providing a common software framework which incorporates all the relevant piece of code and produces output tables, plots and documentation on the web for every new batch of processed data.
One coordinator (SR) and several experts on call.

Current Development

- Automatic tool to check data stability (high PT Leptons)
 - ♦ All code in one common place (cvs)
- TopNt and StNtuple produced shortly after Production data is available: target 4-6 weeks after Production. Turnover rate ~2.5 months.
- The ID code is launched to validate the new ntuples and determine the various efficiencies and SF
 - ♦ Dependency on several tasks:
 - Good Run List
 - Skimmed Data (to avoid large volumes of files)
 - ...
- Output is posted as plots and tables onto PerfIDia web page
- Joint Physics group does the final sign off

Example: Electron ID



PerfIDia

http://ncdf70.fnal.gov:8001/PerfIDia/PerfIDia.html

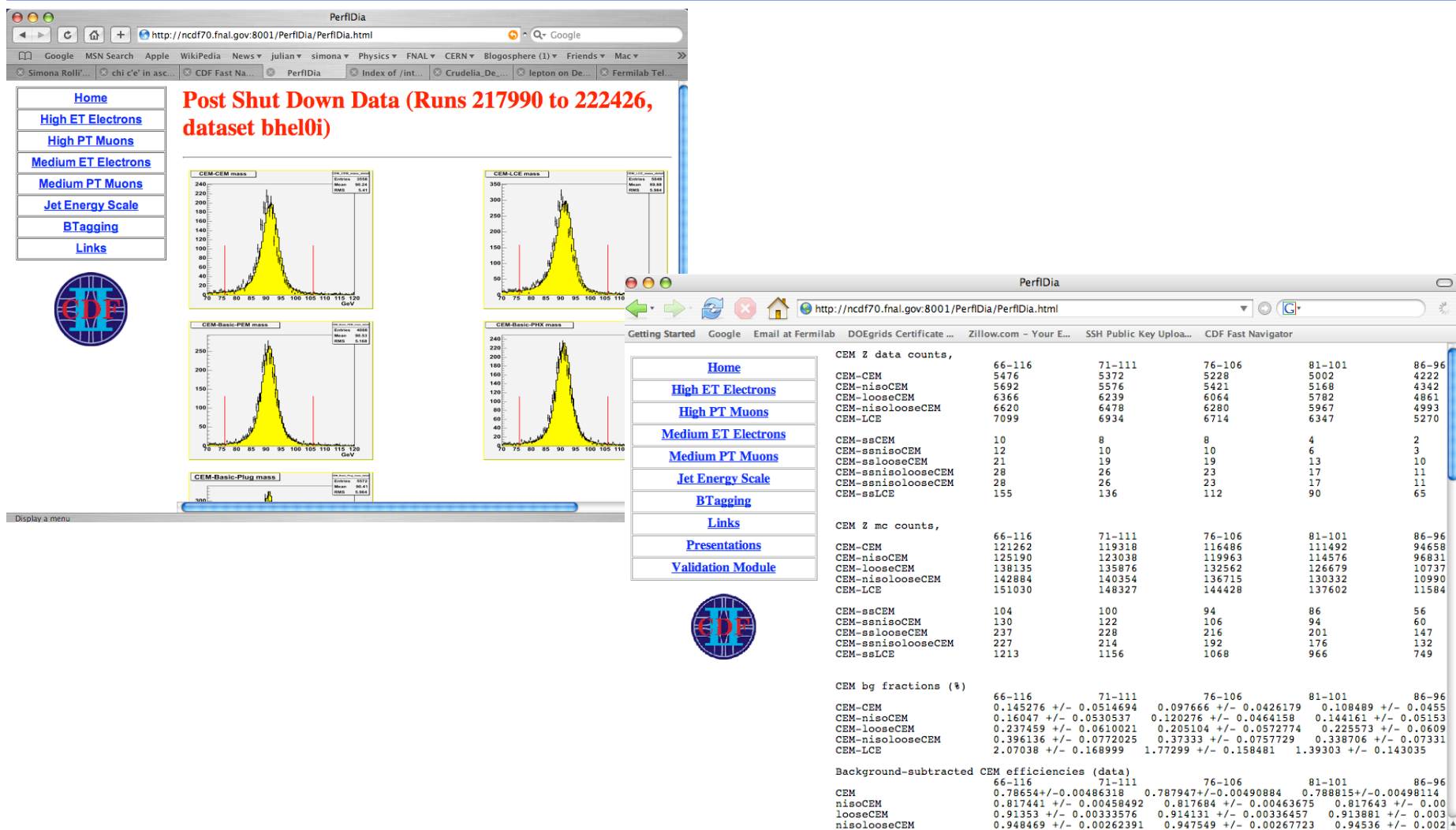
Google Scroogle Apple PerfIDia Tufts Wikipedia News simona julian FNAL CERN Blogosphere (3) References Mac Shopping Travel Physics

Observation of... List of Registe... PerfIDia PPC2008: 2ND ... CDF Fast Navi... chris hays cdf ... Fermilab - Joi... CDF Fast Navi... Chris Ha

- [Home](#)
- [High ET Electrons](#)
- [High PT Muons](#)
- [BMU Muons](#)
- [Jet Energy Scale](#)
- [BTagging](#)
- [Joint Physics SF and Efficiency Spreadsheet](#)
- [Presentations](#)
- [Links](#)
- [Validation Page](#)
- [Old Validation Module](#)

- Documentation:
 - [CDF Note 9148, Dec 2007, P9 to P12 Electron ID and SF](#)
 - [CDF Note 8614, November 27, 2006, 1 fb data](#)
 - [CDF Note 8274, June 1st, 2006, Mesurement Method](#)
- Code:
 - [Instructions on how to run the code](#)
- Results:
 - [Period 17 data: Runs 258880-261005 \(MC sample zetkei\)](#):
 - [Plots](#)
 - [Efficiencies and Scale Factors for data and MC](#)
 - [Period 16 data: Runs 256840-258787 \(MC sample zetkei\)](#):
 - [Plots](#)
 - [Efficiencies and Scale Factors for data and MC](#)
 - [Period 15 data: Runs 254800-256824 \(MC sample zetkei\)](#):
 - [Plots](#)
 - [Efficiencies and Scale Factors for data and MC](#)
 - [Period 14 data: Runs 252836-254686 \(MC sample zetkei\)](#):
 - [Plots](#)
 - [Efficiencies and Scale Factors for data and MC](#)
 - [Period 13 data: Runs 241665-246231 \(MC sample zewke\)](#):
 - [Plots](#)
 - [Efficiencies and Scale Factors for data and MC](#)
 - [Period 12 data: Runs 237850-241664 \(MC sample zewkeh\)](#):
 - [Mass distributions with comparisons between data and MC](#)
 - [Efficiencies and Scale Factors for data and MC \(file not formatted!\)](#)
 - [Efficiencies and Scale Factors for data and MC at large ETA \(eta < 2.8\) \(file not formatted!\)](#)
 - [Period 11 data: Runs 233133 to 237795 \(MC sample zewkcd - newly reprocessed as zewkee\)](#):
 - [Mass distributions with comparisons between data and MC](#)
 - [Efficiencies and Scale Factors for data and MC \(file not formatted!\)](#)
 - [Efficiencies and Scale Factors for data and MC at large ETA \(eta < 2.8\) \(file not formatted!\)](#)
 - [Period 10 data: Runs 228664 to 233111 \(MC sample Zemted\)](#)
 - [Period 9 data : Runs 222529 to 228596 \(MC sample zewkcd - newly reprocessed as zewkmd\)](#):
 - [Mass distributions with comparisons between data and MC](#)
 - [Efficiencies and Scale Factors for data and MC \(file not formatted!\)](#)
 - [Efficiencies and Scale Factors for data and MC at large ETA \(eta < 2.8\) \(file not formatted!\)](#)
 - [Post ShutDown data \(June 2006 to September 2006\): Runs 217990 to 222426 \(MC sample zewkcd\)](#):
 - [Mass distributions with comparisons between data and MC](#)
 - [Efficiencies and Scale Factors for data and MC \(file not formatted!\)](#)

Example: Electron ID



Conclusions

As a “software person”, during the lifecycle of CDFII, I have faced various aspects of software projects:

- ◆ Exposure to new technologies
 - **New languages**, new data access solutions, possibly commercial
- ◆ Involvement in long deliverable projects
 - The **Trigger Simulation** project was a complex project, vital for the rest of the experiment and involving several people at various stages
- ◆ Inventiveness
 - **evtNtuple** was a way to provide early data analyzers with a simple tool, easily portable outside of the more complex, still evolving software framework
- ◆ Help and Support
 - **PerfIDia**

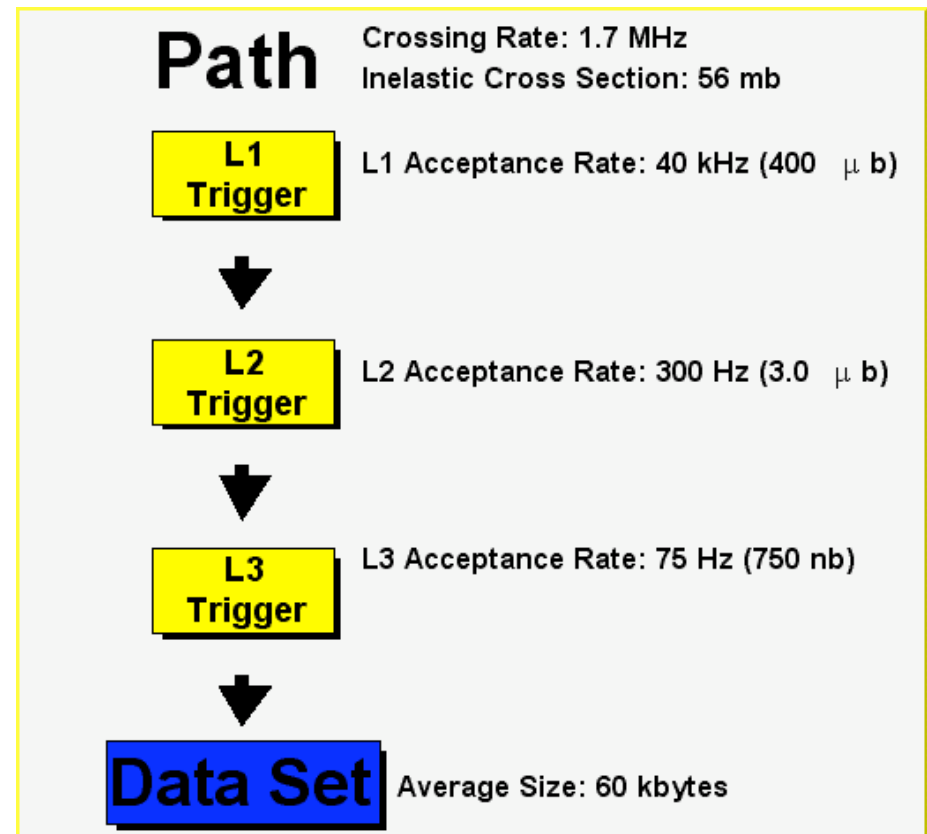
Some more personal conclusions

- I definitely spent the best years of my career at FNAL, from the hustle and bustle of the beginning of CDF Run II to its current *mature* state.
 - ♦ Somebody calls it service work, for me it was an opportunity to learn new things while actually having fun
 - ...the hours spent in the control room during the commissioning run ..
 - ♦ I came to appreciate “technical” work more and more as I found its rewards more tangible than the elusive search for the God’s particle..
 - ...end of the day satisfaction for debugging that nasty line of code...
 - ♦ Sense of team work and accomplishments has been invaluable

Backup Slides

CDF Trigger System

- The trigger is organized in 3 levels
 - L1: hardware, synchronous
 - processing in parallel
 - Pipeline 42 clock cycles deep
 - decision in $5\mu\text{s}$
 - Accept rate max 35 kHz
 - L2: hardware and software, asynchronous
 - Average decision time $30\mu\text{s}$
 - Accept rate max 600 Hz
 - L3: software
 - Farm of PC
 - Speed-optimized offline algorithms
 - Accept rate max 100 Hz





Many of the boards
functionality have been
emulated with simple bit
manipulating routines:

**DIRAC and CrateSum board
have been emulated as from
the online control code**

L1 Calorimeter

Information from the tracker is used in the L1 calorimeters decision. Eight bits per 15° in φ are received through a front panel connector from XFT via the XTRP board -8 bits refer to 8 thresholds in the XTRP board

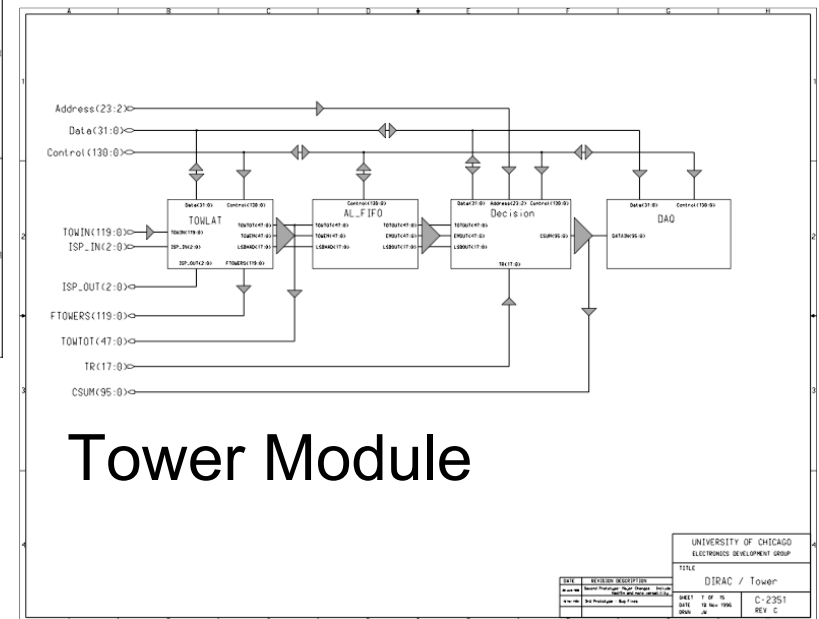
The 8 bits are sent to three SRAMs that decode the information into 3 bits per trigger tower

DIRAC can decode the information regardless of the meaning of the triggers used to setup the thresholds (set the 8 bits)

The same data is sent to each DIRAC card on a ϕ slice. For a crate there are 32 (4x8) bits from XTRP per wedge. The information arrives at a time t_{tracking} after the bunch crossing

The ELPD chips resident on DIRAC are reprogrammable via the ALTERA in-system programmability

Pin	Signal	Pin	Signal
1	Bit 0-	2	Bit 0+
3	Bit 1-	4	Bit 1+
5	Bit 2-	6	Bit 2+
7	Bit 3-	8	Bit 3+
9	Bit 4-	10	Bit 4+
11	Bit 5-	12	Bit 5+
13	Bit 6-	14	Bit 6+
15	Bit 7-	16	Bit 7+



The Tower Module

The 20 bits of energy in each tower come through the custom P3 module with transverse energy and pedestal already subtracted. The 20 bits are divided in 10 bits of EM and 10 bits of HAD. One count corresponds to 125 MeV.

These signals are grouped into a 120 bit bus (**TOWIN: 119:0**) and the bus is fanned out in 6 segments and routed to the **TOWLAT** section within the Tower module.

The TOWLAT contains 6 ELPD chips which latch the input energies at a time $t_{\text{Calorimeter}}$. Each chip represents a trigger tower, receiving 10 bits of EM and 10 bits of HAD data. Ability to mask off a tower is provided (The masked tower will contribute 0 energy)

Via the **FTOWERS(119:0)** bus the two 10-bit energies are sent to L2 boards.

8 of the 10 EM bits are used in a later L1 decision (via **TOWEM(47:0)** bus)

The choice of which 8 bits are used is determined by the state of the Tower Granularity bit in the control registers.

A crude measure of the hadron energy is provided by **LSBHAD(17:0)**, using the 3 or next to three LSBs (choice regulated by another bit in the control registers). These bits are used to make a rough cut on the hadron energy in a tower for electron triggers.

The two 10 bits data words input to TOWLAT are summed into an 8 bit word which is the total energy of the trigger tower (in case of overflow, all the bits are turned on, ie full scale)

TOWTOT(47:0) is the collection of all the 6 chips total energies.

TOWTOT, **TOWEM**, **LSBHAD** are fed to the AL_FIFO section for time alignment with the tracking bits

The output is **TOTOUT**, **EMOUT** and **LSBOUT** are sent to the **Decision** section.

DIRAC Decisions

The outputs from AL_FIFO are sent to the Decision section
Decision also accepts an 18 bits bus (TR(17:0)) from the track section

Twelve (EM+TOT) SRAM's receive the 4 busses and make the trigger decision

For each trigger, one SRAM looks at the TOWTOT and TR and generates the 8 jet trigger decisions. The other SRAM generates the eight electromagnetic trigger decisions. These decisions are combined into the CSUM(95:0) bus.

There are 16 decisions per trigger tower for a total of 96 bits. However there are only 13 L1 trigger bits available for global decision, so the reduction from 16 to 13 is done in the CSUM section.

The CSUM is sent to two modules: CardSum and DAQ.

In DAQ the events are stored while waiting for the L2 decision.